SPAs for performance modelling: Lecture 1 — Introduction

Jane Hillston

LFCS, School of Informatics
The University of Edinburgh
Scotland

8th April 2013

THE UNIVERSITY
of EDINBURGH

Outline

Outline

Performance Modelling

Performance modelling is concerned with the dynamic behaviour of systems and quantified assessment of that behaviour.

Performance Modelling

Performance modelling is concerned with the dynamic behaviour of systems and quantified assessment of that behaviour.

There are often conflicting interests at play:

- Users typically want to optimise external measurements of the dynamics such as response time (as small as possible), throughput (as high as possible) or blocking probability (preferably zero);

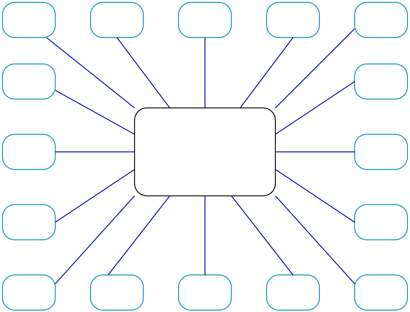
Performance Modelling

Performance modelling is concerned with the dynamic behaviour of systems and quantified assessment of that behaviour.

There are often conflicting interests at play:

- Users typically want to optimise external measurements of the dynamics such as response time (as small as possible), throughput (as high as possible) or blocking probability (preferably zero);
- In contrast, system managers may seek to optimize internal measurements of the dynamics such as utilisation (reasonably high, but not too high), idle time (as small as possible) or failure rates (as low as possible).

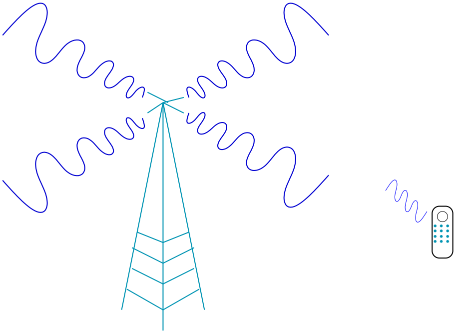
Performance Modelling: Motivation



Capacity planning

- How many clients can the existing server support and maintain reasonable response times?

Performance Modelling: Motivation

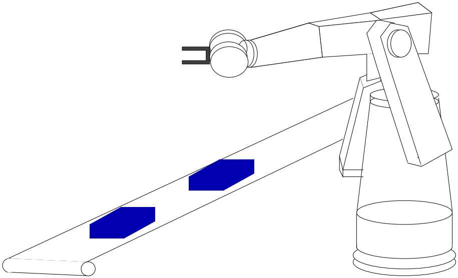


Mobile Telephone Antenna

System Configuration

- How many frequencies do you need to keep blocking probabilities low?

Performance Modelling: Motivation



System Tuning

- What speed of conveyor belt will minimize robot idle time and maximize throughput whilst avoiding lost widgets?

Key notions

A model can be constructed to represent some aspect of the dynamic behaviour of a system.

Key notions

A model can be constructed to represent some aspect of the dynamic behaviour of a system.

Once constructed, such a model becomes a tool with which we can investigate the behaviour of the system.

Modelling computer systems: the challenges

- Physical distance
  - Network latency

Modelling computer systems: the challenges

- Physical distance
  - Network latency
- Partial failures
  - Server may be down
  - Routers may be down

Modelling computer systems: the challenges

- Physical distance
  - Network latency
- Partial failures
  - Server may be down
  - Routers may be down
- Scale
  - Workload characterisation

Modelling computer systems: the challenges

- Physical distance
  - Network latency
- Partial failures
  - Server may be down
  - Routers may be down
- Scale
  - Workload characterisation
- Resource sharing
  - Network contention
  - CPU load

Modelling computer systems: the challenges

- Physical distance — need to represent time
  - Network latency
- Partial failures
  - Server may be down
  - Routers may be down
- Scale
  - Workload characterisation
- Resource sharing
  - Network contention
  - CPU load

Modelling computer systems: the challenges

- Physical distance — need to represent time
  - Network latency
- Partial failures — randomness and probability
  - Server may be down
  - Routers may be down
- Scale
  - Workload characterisation
- Resource sharing
  - Network contention
  - CPU load

Modelling computer systems: the challenges

- Physical distance — need to represent time
  - Network latency
- Partial failures — randomness and probability
  - Server may be down
  - Routers may be down
- Scale — need to quantify population sizes
  - Workload characterisation
- Resource sharing
  - Network contention
  - CPU load

Modelling computer systems: the challenges

- Physical distance — need to represent time
  - Network latency
- Partial failures — randomness and probability
  - Server may be down
  - Routers may be down
- Scale — need to quantify population sizes
  - Workload characterisation
- Resource sharing — need to express percentages
  - Network contention
  - CPU load

Modelling computer systems: the challenges

Time What representation of time will we use?

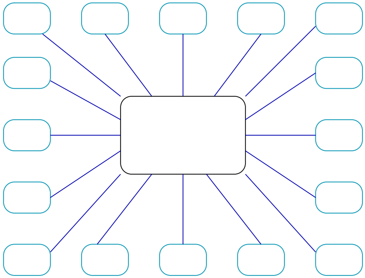
Randomness What kind of random number distributions will we use?

Probability How can we have probabilities in the model without uncertainty in the results?

Scale How can we escape the state-space explosion problem?

Percentages What can it mean to have a fraction of a process?

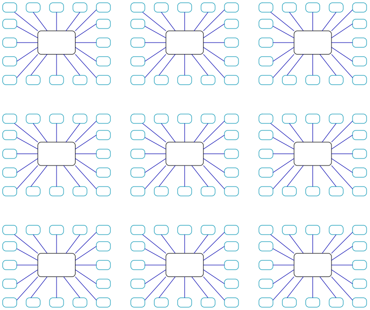
Quantitative Modelling: Motivation



Quality of Service issues

- Can the server maintain reasonable response times?

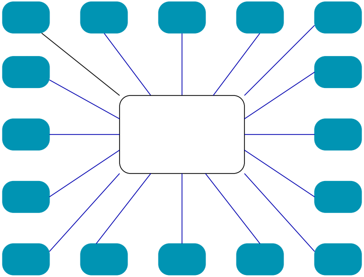
Quantitative Modelling: Motivation



Scalability issues

- How many times do we have to replicate this service to support all of the subscribers?

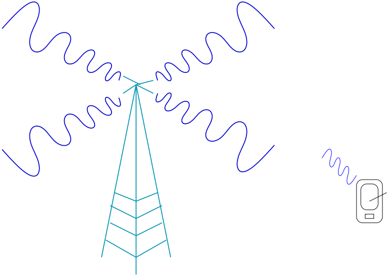
Quantitative Modelling: Motivation



Scalability issues

- Will the server withstand a distributed denial of service attack?

Quantitative Modelling: Motivation



Service-level agreements

- What percentage of downloads do complete within the time we advertised?

Outline

Operational Laws

- Operational laws are simple equations may be regarded as a very abstract model of the average behaviour of almost any system, based on the operational view of the system.

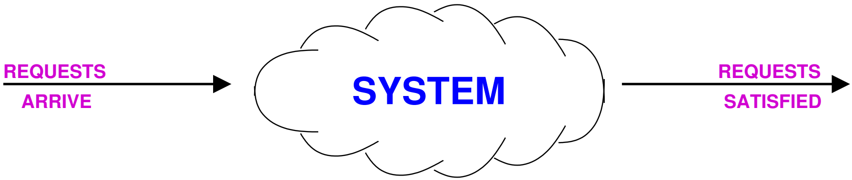
Operational Laws

- Operational laws are simple equations may be regarded as a very abstract model of the average behaviour of almost any system, based on the operational view of the system.
- The laws are very general and make almost no assumptions about the behaviour of the random variables characterising the system.

Operational Laws

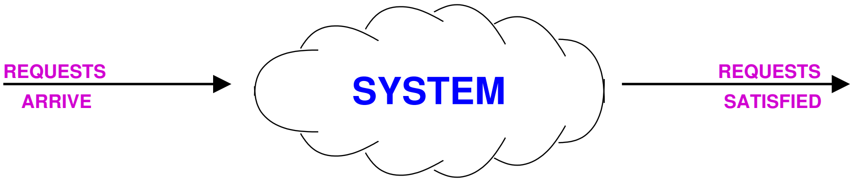
- Operational laws are simple equations may be regarded as a very abstract model of the average behaviour of almost any system, based on the operational view of the system.
- The laws are very general and make almost no assumptions about the behaviour of the random variables characterising the system.
- Another advantage of the laws is their simplicity: this means that they can be applied quickly without detailed knowledge. We will use them sometimes to derive further data from the output observed from models.

Observable variables



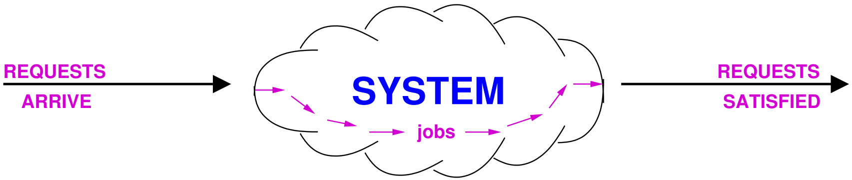
- Operational laws are based on observable variables — values which we could derive from watching a system over a finite period of time.

Observable variables



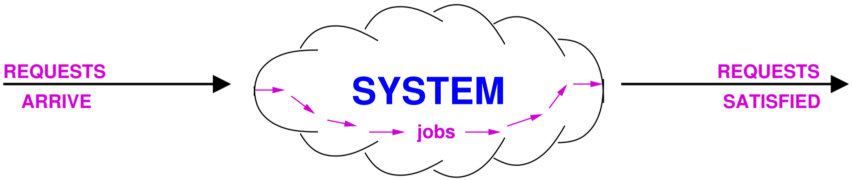
- Operational laws are based on observable variables — values which we could derive from watching a system over a finite period of time.
- We assume that the system receives requests from its environment.

Observable variables



- Operational laws are based on observable variables — values which we could derive from watching a system over a finite period of time.
- We assume that the system receives requests from its environment.
- Each request generates a job or customer within the system.

Observable variables



- Operational laws are based on observable variables — values which we could derive from watching a system over a finite period of time.
- We assume that the system receives requests from its environment.
- Each request generates a job or customer within the system.
- When the job has been processed the system responds to the environment with the completion of the corresponding request.

Observations and measurements

If we observed such an abstract system we might measure the following quantities:

T, the length of time we observe the system;

Observations and measurements

If we observed such an abstract system we might measure the following quantities:

T, the length of time we observe the system;

A, the number of request arrivals we observe;

Observations and measurements

If we observed such an abstract system we might measure the following quantities:

- T, the length of time we observe the system;
- A, the number of request arrivals we observe;
- C, the number of request completions we observe;

Observations and measurements

If we observed such an abstract system we might measure the following quantities:

- T, the length of time we observe the system;
- A, the number of request arrivals we observe;
- C, the number of request completions we observe;
- B, the total amount of time during which the system is busy ($B \leq T$);

Observations and measurements

If we observed such an abstract system we might measure the following quantities:

- T, the length of time we observe the system;
- A, the number of request arrivals we observe;
- C, the number of request completions we observe;
- B, the total amount of time during which the system is busy ($B \leq T$);
- N, the average number of jobs in the system.

Four important quantities

From these observed values we can derive the following four important quantities:

$\lambda = A/T$, the arrival rate;

Four important quantities

From these observed values we can derive the following four important quantities:

$\lambda = A/T$, the arrival rate;

$X = C/T$, the throughput or completion rate,

Four important quantities

From these observed values we can derive the following four important quantities:

$\lambda = A/T$, the arrival rate;

$X = C/T$, the throughput or completion rate,

$U = B/T$, the utilisation;

Four important quantities

From these observed values we can derive the following four important quantities:

$$\lambda = A/T, \text{ the arrival rate;}$$

$$X = C/T, \text{ the throughput or completion rate,}$$

$$U = B/T, \text{ the utilisation;}$$

$$S = B/C, \text{ the mean service time per completed job.}$$

Job flow balance

- We will assume that the system is job flow balanced. This means that the number of arrivals is equal to the number of completions during an observation period, i.e. $A = C$.

Job flow balance

- We will assume that the system is job flow balanced. This means that the number of arrivals is equal to the number of completions during an observation period, i.e. $A = C$.
- This is a testable assumption because an analyst can always test whether the assumption holds.

Job flow balance

- We will assume that the system is job flow balanced. This means that the number of arrivals is equal to the number of completions during an observation period, i.e. $A = C$.
- This is a testable assumption because an analyst can always test whether the assumption holds.
- Note that if the system is job flow balanced the arrival rate will be the same as the completion rate, that is, $\lambda = X$.

Little's Law

Little's Law

$$N = XW$$

The average number of jobs in a system is equal to the product of the throughput of the system and the average time spent in that system by a job.

Example

Consider a disk that serves 40 requests/second ($X = 40$) and suppose that on average there are 4 requests present in the disk system (waiting to be served or in service) ($N = 4$).

Example

Consider a disk that serves 40 requests/second ($X = 40$) and suppose that on average there are 4 requests present in the disk system (waiting to be served or in service) ($N = 4$).

Little's law tells us that the average time spent at the disk by a request must be $4/40 = 0.1$ seconds.

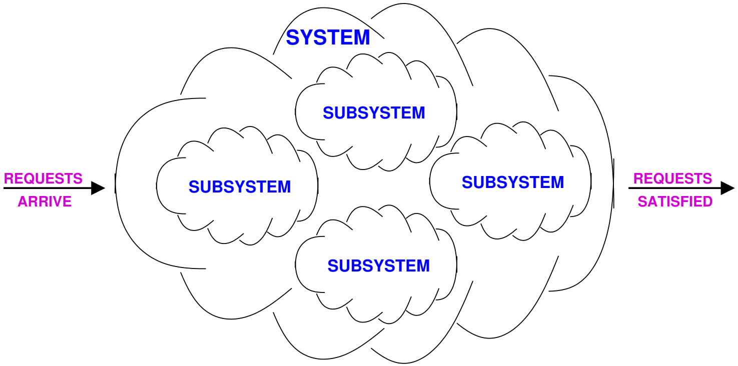
Example

Consider a disk that serves 40 requests/second ($X = 40$) and suppose that on average there are 4 requests present in the disk system (waiting to be served or in service) ($N = 4$).

Little's law tells us that the average time spent at the disk by a request must be $4/40 = 0.1$ seconds.

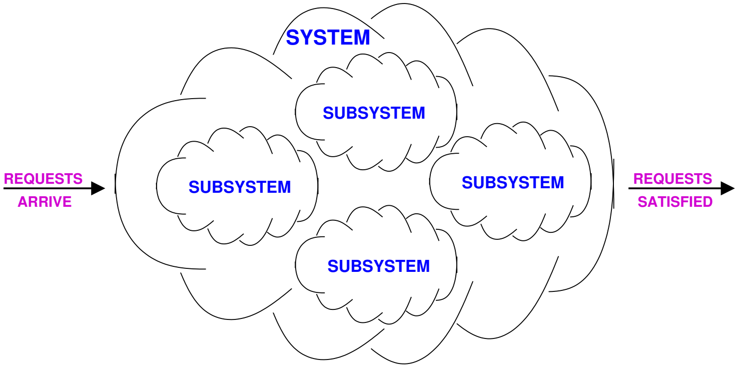
If we know that each request requires 0.0225 seconds of disk service we can then deduce that the average queueing time is 0.0775 seconds.

Subsystems within Systems



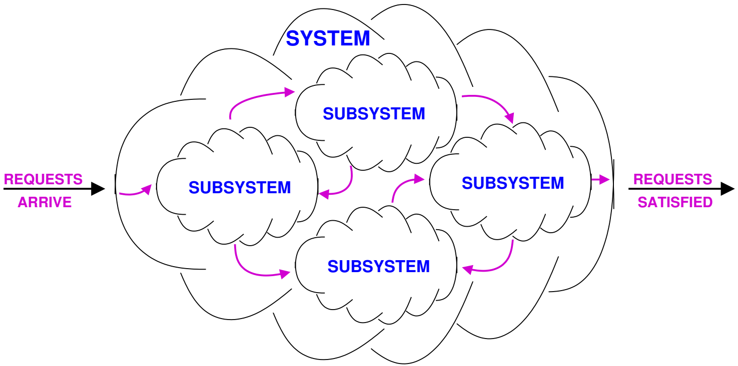
- A system may be regarded as being made up of a number of devices or resources.

Subsystems within Systems



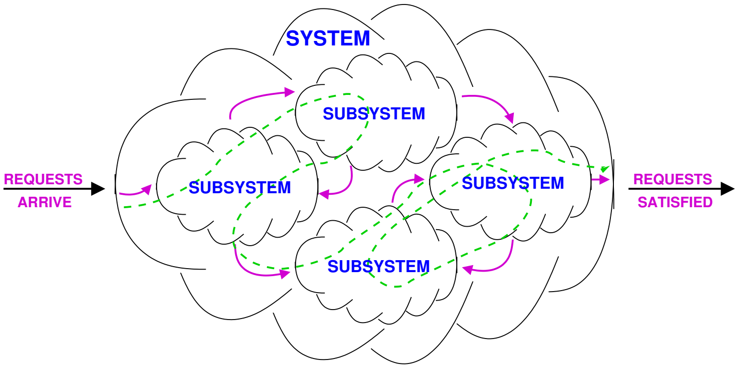
- A system may be regarded as being made up of a number of devices or resources.
- Each of these may be treated as a system in its own right from the perspective of operational laws.

Subsystems within Systems



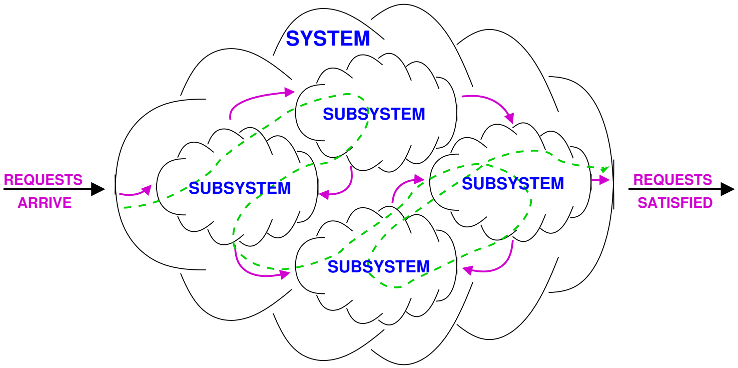
An external request generates a job within the system; this job may then circulate between the resources until all necessary processing has been done; as it arrives at each resource it is treated as a request, generating a job internal to that resource.

Subsystems within Systems



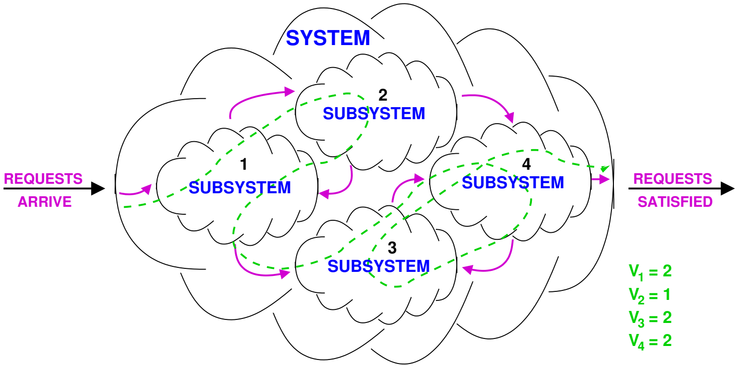
An external request generates a job within the system; this job may then circulate between the resources until all necessary processing has been done; as it arrives at each resource it is treated as a request, generating a job internal to that resource.

Visit count



In an observation interval we can count not only completions external to the system, but also the number of completions at each resource within the system.

Visit count



We define the visit count, V_i, of the ith resource to be the ratio of the number of completions at that resource to the number of system completions $V_i \equiv C_i/C$.

Visit count: example

For example, if, during an observation interval, we measure 10 system completions and 150 completions at a specific disk, then on the average each system-level request requires 15 disk operations.

Forced Flow Law

The forced flow law captures the relationship between the different components within a system. It states that the throughputs or flows, in all parts of a system must be proportional to one another.

Forced Flow Law

The forced flow law captures the relationship between the different components within a system. It states that the throughputs or flows, in all parts of a system must be proportional to one another.

Forced Flow Law

$$X_i = XV_i$$

The throughput at the ith resource is equal to the product of the throughput of the system and the visit count at that resource.

Example

- Consider a robotic workcell within a computerised manufacturing system which processes widgets.

Example

- Consider a robotic workcell within a computerised manufacturing system which processes widgets.
- Suppose that processing each widget requires 4 accesses to the lathe and 2 accesses to the press.

Example

- Consider a robotic workcell within a computerised manufacturing system which processes widgets.
- Suppose that processing each widget requires 4 accesses to the lathe and 2 accesses to the press.
- We know that the lathe processes 8 widgets in a minute and we want to know the throughput of the press.

Example

- Consider a robotic workcell within a computerised manufacturing system which processes widgets.
- Suppose that processing each widget requires 4 accesses to the lathe and 2 accesses to the press.
- We know that the lathe processes 8 widgets in a minute and we want to know the throughput of the press.
- The throughput of the workcell will be proportional to the lathe throughput, i.e. $X = X_{lathe}/V_{lathe} = 8/4 = 2$.

Example

- Consider a robotic workcell within a computerised manufacturing system which processes widgets.
- Suppose that processing each widget requires 4 accesses to the lathe and 2 accesses to the press.
- We know that the lathe processes 8 widgets in a minute and we want to know the throughput of the press.
- The throughput of the workcell will be proportional to the lathe throughput, i.e. $X = X_{lathe}/V_{lathe} = 8/4 = 2$.
- The throughput of the press will be $X_{press} = X \times V_{press} = 2 \times 2 = 4$.

Example

- Consider a robotic workcell within a computerised manufacturing system which processes widgets.
- Suppose that processing each widget requires 4 accesses to the lathe and 2 accesses to the press.
- We know that the lathe processes 8 widgets in a minute and we want to know the throughput of the press.
- The throughput of the workcell will be proportional to the lathe throughput, i.e. $X = X_{lathe}/V_{lathe} = 8/4 = 2$.
- The throughput of the press will be $X_{press} = X \times V_{press} = 2 \times 2 = 4$.
- Thus the press throughput is 4 widgets per minute.

Utilisation Law

- If we know the amount of processing each job requires at a resource then we can calculate the utilisation of the resource.

Utilisation Law

- If we know the amount of processing each job requires at a resource then we can calculate the utilisation of the resource.
- Let us assume that each time a job visits the ith resource the amount of processing, or service time it requires is S_i.

Utilisation Law

- If we know the amount of processing each job requires at a resource then we can calculate the utilisation of the resource.
- Let us assume that each time a job visits the ith resource the amount of processing, or service time it requires is S_i.
- (Note that service time is not necessarily the same as the residence time of the job at that resource: in general a job might have to wait for some time before processing begins.)

Utilisation Law

- If we know the amount of processing each job requires at a resource then we can calculate the utilisation of the resource.
- Let us assume that each time a job visits the ith resource the amount of processing, or service time it requires is S_i.
- (Note that service time is not necessarily the same as the residence time of the job at that resource: in general a job might have to wait for some time before processing begins.)
- The total amount of service that a system job generates at the ith resource is called the service demand, D_i:

$$D_i = S_i V_i$$

Utilisation Law

The utilisation of a resource, the percentage of time that the ith resource is in use processing to a job, is denoted U_i.

Utilisation Law

$$U_i = X_i S_i = X D_i$$

The utilisation of a resource is equal to the product of the throughput of that resource and the average service requirement at that resource.

Utilisation Example

- Consider again the disk that is serving 40 requests/second, each of which requires 0.0225 seconds of disk service.

Utilisation Example

- Consider again the disk that is serving 40 requests/second, each of which requires 0.0225 seconds of disk service.
- The utilisation law tells us that the utilisation of the disk must be $40 \times 0.0225 = 90\%$.

Interactive Response Time Law

- Back when most processing was done on shared mainframes, think time, Z, was quite literally the length of time that a programmer spent thinking before submitting another job.

Interactive Response Time Law

- Back when most processing was done on shared mainframes, think time, Z, was quite literally the length of time that a programmer spent thinking before submitting another job.
- More generally in interactive systems, jobs spend time in the system not engaged in processing, or waiting for processing: this may be because of interaction with a human user, or may be for some other reason.

Interactive Response Time Law

- Back when most processing was done on shared mainframes, think time, Z, was quite literally the length of time that a programmer spent thinking before submitting another job.
- More generally in interactive systems, jobs spend time in the system not engaged in processing, or waiting for processing: this may be because of interaction with a human user, or may be for some other reason.
- The key feature of such a system is that the residence time can no longer be taken as a true reflection of the response time of the system.

Example

- For example, if we are studying a cluster of workstations with a central file server to investigate the load on the file server, the think time might represent the average time that each workstation spends processing locally without access to the file server.

Example

- For example, if we are studying a cluster of workstations with a central file server to investigate the load on the file server, the think time might represent the average time that each workstation spends processing locally without access to the file server.
- At the end of this non-processing period the job generates a fresh request.

Think time, residence time, response time

- The think time represents the time between processing being completed and the job becoming available as a request again.

Think time, residence time, response time

- The think time represents the time between processing being completed and the job becoming available as a request again.
- Thus the residence time of the job, as calculated by Little's Law as the time from arrival to completion, is greater than the system's response time.

Interactive Response Time Law

The interactive response time law reflects this: it calculates the response time, R as follows:

Interactive Response Time Law

$$R = N/X - Z$$

The response time in an interactive system is the residence time minus the think time.

Interactive Response Time Law

The interactive response time law reflects this: it calculates the response time, R as follows:

Interactive Response Time Law

$$R = N/X - Z$$

The response time in an interactive system is the residence time minus the think time.

Note that if the think time is zero, $Z = 0$ and $R = W$, then the interactive response time law simply becomes Little's Law.

Interactive Response Time Law: Example

- Suppose that the library catalogue system has 64 interactive users connected via Browsers, that the average think time is 30 seconds, and that system throughput is 2 interactions/second.

Interactive Response Time Law: Example

- Suppose that the library catalogue system has 64 interactive users connected via Browsers, that the average think time is 30 seconds, and that system throughput is 2 interactions/second.
- Then the interactive response time law tells us that the response time must be $64/2 - 30 = 2$ seconds.

Outline

The discrete event view

In general we wish to have a more detailed (mechanistic) view of the system under study than that presented by the operational laws. In this course we will consider discrete event systems.

The discrete event view

In general we wish to have a more detailed (mechanistic) view of the system under study than that presented by the operational laws. In this course we will consider discrete event systems.

The state of the system is characterised by variables which take distinct values and which change by discrete events, i.e. at a distinct time something happens within the system which results in a change in one or more of the state variables.

The discrete event view: example

We might be interested in the number of nodes in a communication network which are currently waiting to send a message N.

- If a node, which was not previously waiting, generates a message and is now waiting to send then $N \to N + 1$, or
- If a node, which was previously waiting, successfully transmits its message then $N \to N - 1$.

Discrete time vs. Continuous time

Within discrete event systems there is a distinction between a discrete time representation and a continuous time representation:

Discrete time vs. Continuous time

Within discrete event systems there is a distinction between a discrete time representation and a continuous time representation:

Discrete time: such models only consider the system at predetermined moments in time, which are typically evenly spaced, eg. at each clock "tick".

Continuous time: such models consider the system at the time of each event so the time parameter in such models is conceptually continuous.

Discrete time vs. Continuous time

Within discrete event systems there is a distinction between a discrete time representation and a continuous time representation:

Discrete time: such models only consider the system at predetermined moments in time, which are typically evenly spaced, eg. at each clock "tick".

Continuous time: such models consider the system at the time of each event so the time parameter in such models is conceptually continuous.

At levels of abstraction above the hardware clock continuous time models are generally appropriate for computer and communication systems.

Quantitative modelling

When systems are modelled to verify their functional behaviour (correctness), all definite values are abstracted away — qualitative modelling.

Quantitative modelling

When systems are modelled to verify their functional behaviour (correctness), all definite values are abstracted away — qualitative modelling.

In contrast, performance modelling is quantitative modelling as we must take into account explicit values for time (latency, service time etc.) and probability (choices, alternative outcomes, mixed workload).

Quantitative modelling

When systems are modelled to verify their functional behaviour (correctness), all definite values are abstracted away — qualitative modelling.

In contrast, performance modelling is quantitative modelling as we must take into account explicit values for time (latency, service time etc.) and probability (choices, alternative outcomes, mixed workload).

Probability will be used to represent randomness (e.g. from human users) but also as an abstraction over unknown values (e.g. service times).

Outline

Random experiments and events

- To apply probability theory to the process under study, we view it as a random experiment.

Random experiments and events

- To apply probability theory to the process under study, we view it as a random experiment.
- The sample space of a random experiment is the set of all individual outcomes of the experiment.

Random experiments and events

- To apply probability theory to the process under study, we view it as a random experiment.
- The sample space of a random experiment is the set of all individual outcomes of the experiment.
- These individual outcomes are also called sample points or elementary events.

Random experiments and events

- To apply probability theory to the process under study, we view it as a random experiment.
- The sample space of a random experiment is the set of all individual outcomes of the experiment.
- These individual outcomes are also called sample points or elementary events.
- An event is a subset of a sample space.

Random variables

We are interested in the dynamics of a system as events happen over time.

A function which associates a (real-valued) number with the outcome of an experiment is known as a random variable.

Formally, a random variable X is a real-valued function defined on a sample space Ω.

Measurable functions

If X is a random variable, and x is a real number, we write $X \leq x$ for the event

$$\{\, \omega : \omega \in \Omega \text{ and } X(\omega) \leq x \,\}$$

Measurable functions

If X is a random variable, and x is a real number, we write $X \leq x$ for the event

$$\{\, \omega : \omega \in \Omega \text{ and } X(\omega) \leq x \,\}$$

and we write $X = x$ for the event

$$\{\, \omega : \omega \in \Omega \text{ and } X(\omega) = x \,\}$$

Measurable functions

If X is a random variable, and x is a real number, we write $X \leq x$ for the event

$$\{\, \omega : \omega \in \Omega \text{ and } X(\omega) \leq x \,\}$$

and we write $X = x$ for the event

$$\{\, \omega : \omega \in \Omega \text{ and } X(\omega) = x \,\}$$

We require that for a random variable X, the set $X \leq x$ is an event for each real x. This is necessary so that probability calculations can be made. A function having this property is said to be a measurable function or measurable in the Borel sense.

Distribution function

For each random variable X we define its distribution function F for each real x by

$$F(x) = \Pr[X \leq x]$$

Distribution function

For each random variable X we define its distribution function F for each real x by

$$F(x) = \Pr[X \leq x]$$

We associate another function $p(\cdot)$, called the probability mass function, with X (pmf), for each x:

$$p(x) = \Pr[X = x]$$

Distribution function

For each random variable X we define its distribution function F for each real x by

$$F(x) = \Pr[X \leq x]$$

We associate another function $p(\cdot)$, called the probability mass function, with X (pmf), for each x:

$$p(x) = \Pr[X = x]$$

A random variable X is continuous if $p(x) = 0$ for all real x.

Distribution function

For each random variable X we define its distribution function F for each real x by

$$F(x) = \Pr[X \leq x]$$

We associate another function $p(\cdot)$, called the probability mass function, with X (pmf), for each x:

$$p(x) = \Pr[X = x]$$

A random variable X is continuous if $p(x) = 0$ for all real x.

(If X is a continuous random variable, then X can assume infinitely many values, and so it is reasonable that the probability of its assuming any specific value we choose beforehand is zero.)

Exponential random variables, distribution function

The random variable X is said to be an exponential random variable with parameter λ ($\lambda > 0$) or to have an exponential distribution with parameter λ if it has the distribution function

$$F(x) = \begin{cases} 1 - e^{-\lambda x} & \text{for } x > 0 \\ 0 & \text{for } x \leq 0 \end{cases}$$

Exponential random variables, distribution function

The random variable X is said to be an exponential random variable with parameter λ ($\lambda > 0$) or to have an exponential distribution with parameter λ if it has the distribution function

$$F(x) = \begin{cases} 1 - e^{-\lambda x} & \text{for } x > 0 \\ 0 & \text{for } x \leq 0 \end{cases}$$

Some authors call this distribution the negative exponential distribution.

Exponential random variables, density function

The density function $f = dF/dx$ is given by

$$f(x) = \begin{cases} \lambda e^{-\lambda x} & \text{if } x > 0 \\ 0 & \text{if } x \leq 0 \end{cases}$$

Mean, or expected value

If X is a continuous random variable with density function $f(\cdot)$, we define the mean or expected value of X, $\mu = E[X]$ by

$$\mu = E[X] = \int_{-\infty}^{\infty} x f(x) dx$$

Mean, or expected value

If X is a continuous random variable with density function $f(\cdot)$, we define the mean or expected value of X, $\mu = E[X]$ by

$$\mu = E[X] = \int_{-\infty}^{\infty} x f(x) dx$$

If X is a discrete random variable with probability mass function $p(\cdot)$, we define the mean or expected value of $X \in S$, $\mu = E[X]$ by

$$E(X) = \sum_{x \in S} x p(x)$$

Mean, or expected value, of the exponential distribution

Suppose X has an exponential distribution with parameter $\lambda > 0$.

Then

$$\mu = E[X] = \int_{-\infty}^{\infty} x\lambda e^{-\lambda x} dx = \frac{1}{\lambda}$$

Exponential inter-event time distribution

The time interval between successive events can also be deduced.

Let $F(t)$ be the distribution function of T, the time between events. Consider $\Pr(T > t) = 1 - F(t)$:

$$\begin{aligned}\Pr(T > t) &= \Pr(\text{No events in an interval of length } t) \\ &= 1 - F(t) \\ &= 1 - (1 - e^{-\lambda t}) \\ &= e^{-\lambda t}\end{aligned}$$

Memoryless property of the exponential distribution

The memoryless property of the exponential distribution is so called because the time to the next event is independent of when the last event occurred.

Memoryless property of the exponential distribution

Suppose that the last event was at time 0. What is the probability that the next event will be after $t + s$, given that time t has elapsed since the last event, and no events have occurred?

Memoryless property of the exponential distribution

Suppose that the last event was at time 0. What is the probability that the next event will be after $t + s$, given that time t has elapsed since the last event, and no events have occurred?

$$
\begin{aligned}
\Pr(T > t + s \mid T > t) &= \frac{\Pr(T > t + s \text{ and } T > t)}{\Pr(T > t)} \\
&= \frac{e^{-\lambda(t+s)}}{e^{-\lambda t}} \\
&= e^{-\lambda s}
\end{aligned}
$$

Memoryless property of the exponential distribution

Suppose that the last event was at time 0. What is the probability that the next event will be after $t+s$, given that time t has elapsed since the last event, and no events have occurred?

$$\begin{aligned}
\Pr(T > t+s \mid T > t) &= \frac{\Pr(T > t+s \text{ and } T > t)}{\Pr(T > t)} \\
&= \frac{e^{-\lambda(t+s)}}{e^{-\lambda t}} \\
&= e^{-\lambda s}
\end{aligned}$$

This value is independent of t (and so the time already spent has not been remembered).

Outline

Stochastic Process

- Formally, a stochastic model is one represented as a stochastic process;

Stochastic Process

- Formally, a stochastic model is one represented as a stochastic process;
- A stochastic process is a set of random variables $\{X(t), t \in T\}$.

Stochastic Process

- Formally, a stochastic model is one represented as a stochastic process;
- A stochastic process is a set of random variables$\{X(t), t \in T\}$.
- T is called the index set usually taken to represent time.

Stochastic Process

- Formally, a stochastic model is one represented as a stochastic process;
- A stochastic process is a set of random variables $\{X(t), t \in T\}$.
- T is called the index set usually taken to represent time.
- Since we consider continuous time models $T = \mathbb{R}^{\geq 0}$, the set of non-negative real numbers.

State Space

The state space of a stochastic process is the set of all possible values that the random variables $X(t)$ can assume.

State Space

The state space of a stochastic process is the set of all possible values that the random variables $X(t)$ can assume.

Each of these values is called a state of the process.

State Space

The state space of a stochastic process is the set of all possible values that the random variables $X(t)$ can assume.

Each of these values is called a state of the process.

Any set of instances of $\{X(t), t \in T\}$ can be regarded as a path of a particle moving randomly in the state space, S, its position at time t being $X(t)$.

State Space

The state space of a stochastic process is the set of all possible values that the random variables $X(t)$ can assume.

Each of these values is called a state of the process.

Any set of instances of $\{X(t), t \in T\}$ can be regarded as a path of a particle moving randomly in the state space, S, its position at time t being $X(t)$.

These paths are called sample paths or realisations of the stochastic process.

Properties of Stochastic Processes

In this course we will focus on stochastic processes with the following properties:

Properties of Stochastic Processes

In this course we will focus on stochastic processes with the following properties:

$\{X(t)\}$ **is a Markov process**.

This implies that $\{X(t)\}$ has the Markov or memoryless property: given the value of $X(t)$ at some time $t \in T$, the future path $X(s)$ for $s > t$ does not depend on knowledge of the past history $X(u)$ for $u < t$, i.e. for $t_1 < \cdots < t_n < t_{n+1}$,

$$\Pr(X(t_{n+1}) = x_{n+1} \mid X(t_n) = x_n, \ldots, X(t_1) = x_1) = \Pr(X(t_{n+1}) = x_{n+1} \mid X(t_n) = x_n)$$

Properties of Stochastic Processes

In this course we will focus on stochastic processes with the following properties:

$\{X(t)\}$ **is irreducible**.

This implies that all states in S can be reached from all other states, by following the transitions of the process. If we draw a directed graph of the state space with a node for each state and an arc for each event, or transition, then for any pair of nodes there is a path connecting them, i.e. the graph is strongly connected.

Properties of Stochastic Processes

In this course we will focus on stochastic processes with the following properties:

$\{X(t)\}$ **is stationary**:

for any $t_1, \ldots t_n \in T$ and $t_1 + \tau, \ldots, t_n + \tau \in T$ $(n \geq 1)$, then the process's joint distributions are unaffected by the change in the time axis and so,

$$F_{X(t_1+\tau)\ldots X(t_n+\tau)} = F_{X(t_1)\ldots X(t_n)}$$

Properties of Stochastic Processes

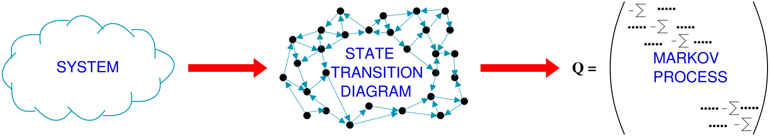
In this course we will focus on stochastic processes with the following properties:

$\{X(t)\}$ **is time homogeneous**:

the behaviour of the system does not depend on when it is observed. In particular, the transition rates between states are independent of the time at which the transitions occur. Thus, for all t and s, it follows that

$$\Pr(X(t+\tau) = x_k \mid X(t) = x_j) = \Pr(X(s+\tau) = x_k \mid X(s) = x_j).$$

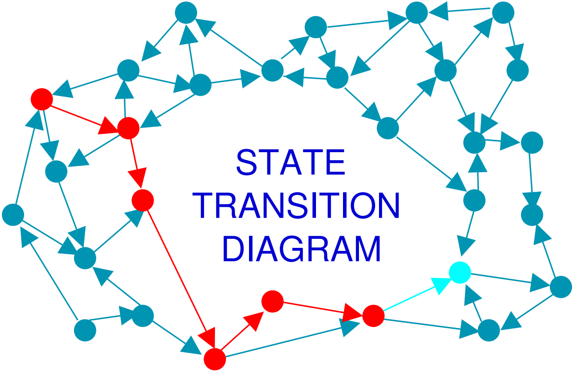
Performance Modelling using CTMC



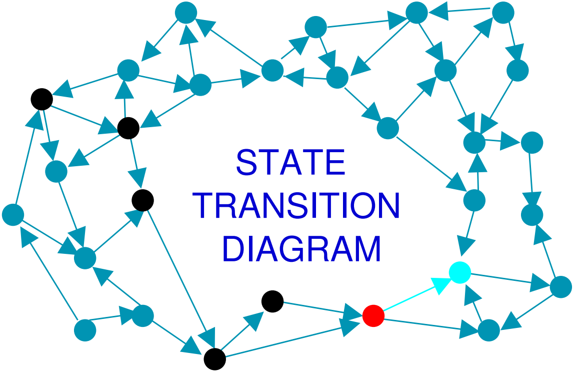
A stochastic process $X(t)$ is a Markov process iff for all $t_0 < t_1 < \ldots < t_n < t_{n+1}$, the joint probability distribution of $(X(t_0), X(t_1), \ldots, X(t_n), X(t_{n+1}))$ is such that

$$\Pr(X(t_{n+1}) = s_{i_{n+1}} \mid X(t_0) = s_{i_0}, \ldots, X(t_n) = s_{i_n}) = \Pr(X(t_{n+1}) = s_{i_{n+1}} \mid X(t_n) = s_{i_n})$$

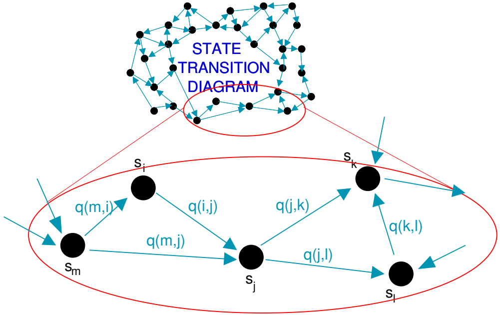
Performance Modelling using CTMC



Performance Modelling using CTMC

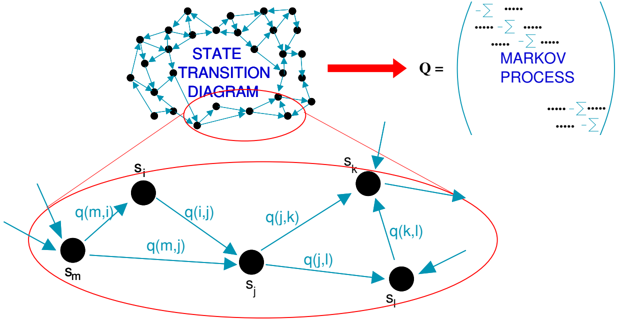


Performance Modelling using CTMC



A negative exponentially distributed duration is associated with each transition.

Performance Modelling using CTMC



these parameters form the entries of the infinitesimal generator matrix Q

Exit rate and sojourn time

In any stochastic process the time spent in a state is called the sojourn time.

Exit rate and sojourn time

In any stochastic process the time spent in a state is called the sojourn time.

In a Markov process the rate of leaving a state x_i, q_i the exit rate, is exponentially distributed with the rate which is the sum of all the individual transitions that leave the state, i.e. $q_i = \sum_{j=1, j\neq i}^{N} q_{i,j}$.

Exit rate and sojourn time

In any stochastic process the time spent in a state is called the sojourn time.

In a Markov process the rate of leaving a state x_i, q_i the exit rate, is exponentially distributed with the rate which is the sum of all the individual transitions that leave the state, i.e. $q_i = \sum_{j=1, j \neq i}^{N} q_{i,j}$.

This follow from the superposition principle of exponential distributions.

Exit rate and sojourn time

In any stochastic process the time spent in a state is called the sojourn time.

In a Markov process the rate of leaving a state x_i, q_i the exit rate, is exponentially distributed with the rate which is the sum of all the individual transitions that leave the state, i.e. $q_i = \sum_{j=1, j \neq i}^{N} q_{i,j}$.

This follow from the superposition principle of exponential distributions.

It follows that the sojourn time will be $1/q_i$.

Exit rate and sojourn time

In any stochastic process the time spent in a state is called the sojourn time.

In a Markov process the rate of leaving a state x_i, q_i the exit rate, is exponentially distributed with the rate which is the sum of all the individual transitions that leave the state, i.e. $q_i = \sum_{j=1, j \neq i}^{N} q_{i,j}$.

This follow from the superposition principle of exponential distributions.

It follows that the sojourn time will be $1/q_i$.

Note that it follows from the Markov property that sojourn times are memoryless.

Transition rates and transition probabilities

At time τ, the probability that there is a state transition in the interval $(\tau, \tau + dt)$ is $q_i dt + o(dt)$.

Transition rates and transition probabilities

At time τ, the probability that there is a state transition in the interval $(\tau, \tau + dt)$ is $q_i dt + o(dt)$.

When a transition out of state x_i occurs, the new state is x_j with probability p_{ij}, which must depend only on i and j (Markov).

Transition rates and transition probabilities

At time τ, the probability that there is a state transition in the interval $(\tau, \tau + dt)$ is $q_i dt + o(dt)$.

When a transition out of state x_i occurs, the new state is x_j with probability p_{ij}, which must depend only on i and j (Markov).

Thus, for $i \neq j, i, j \in S$,
$\Pr(X(\tau + dt) = j \mid X(\tau) = i) = q_{ij} dt + o(dt)$
where the $q_{ij} = q_i p_{ij}$, by the decomposition property.

Transition rates and transition probabilities

At time τ, the probability that there is a state transition in the interval $(\tau, \tau + dt)$ is $q_i dt + o(dt)$.

When a transition out of state x_i occurs, the new state is x_j with probability p_{ij}, which must depend only on i and j (Markov).

Thus, for $i \neq j, i, j \in S$,
$\Pr(X(\tau + dt) = j \mid X(\tau) = i) = q_{ij} dt + o(dt)$
where the $q_{ij} = q_i p_{ij}$, by the decomposition property.

The q_{ij} are called the instantaneous transition rates.

Transition rates and transition probabilities

At time τ, the probability that there is a state transition in the interval $(\tau, \tau + dt)$ is $q_i dt + o(dt)$.

When a transition out of state x_i occurs, the new state is x_j with probability p_{ij}, which must depend only on i and j (Markov).

Thus, for $i \neq j, i, j \in S$,
$\Pr(X(\tau + dt) = j \mid X(\tau) = i) = q_{ij} dt + o(dt)$
where the $q_{ij} = q_i p_{ij}$, by the decomposition property.

The q_{ij} are called the instantaneous transition rates.

The transition probability p_{ij} is the probability, given that a transition out of state i occurs, that it is the transition to state j. By the definition of conditional probability, this is $p_{ij} = q_{ij}/q_i$.

Infinitesimal Generator Matrix

The state transition diagram of a Markov process captures all the information about the states of the system and the transitions which can occur between then.

Infinitesimal Generator Matrix

The state transition diagram of a Markov process captures all the information about the states of the system and the transitions which can occur between then.

When we are reasoning about the process we find it convenient to capture this information in a matrix, $\mathbf{Q}$, termed the infinitesimal generator matrix.

Infinitesimal Generator Matrix

The state transition diagram of a Markov process captures all the information about the states of the system and the transitions which can occur between then.

When we are reasoning about the process we find it convenient to capture this information in a matrix, $\mathbf{Q}$, termed the infinitesimal generator matrix.

For a state space of size N, this is a $N \times N$ matrix, where entry $q(i,j)$ or $q_{i,j}$, records the transition rate of moving from state x_i to state x_j.

Infinitesimal Generator Matrix

The state transition diagram of a Markov process captures all the information about the states of the system and the transitions which can occur between then.

When we are reasoning about the process we find it convenient to capture this information in a matrix, $\mathbf{Q}$, termed the infinitesimal generator matrix.

For a state space of size N, this is a $N \times N$ matrix, where entry $q(i,j)$ or $q_{i,j}$, records the transition rate of moving from state x_i to state x_j.

By convention, the diagonal entries $q_{i,i}$ are the negative row sum for row i, i.e.

$$q_{i,i} = \sum_{j=1, j \neq i}^{N} q_{i,j}$$

Steady state probability distribution

In performance modelling we are often interested in the probability distribution of the random variable $X(t)$ over the state space S, as the system settles into a regular pattern of behaviour.

Steady state probability distribution

In performance modelling we are often interested in the probability distribution of the random variable $X(t)$ over the state space S, as the system settles into a regular pattern of behaviour.

This is termed the steady state probability distribution.

Steady state probability distribution

In performance modelling we are often interested in the probability distribution of the random variable $X(t)$ over the state space S, as the system settles into a regular pattern of behaviour.

This is termed the steady state probability distribution.

From this probability distribution we will derive performance measures based on subsets of states where some condition holds.

Existence of a steady state probability distribution

For every time-homogeneous, finite, irreducible Markov process with state space S, there exists a steady state probability distribution

$$\{\pi_k, x_k \in S\}$$

Existence of a steady state probability distribution

For every time-homogeneous, finite, irreducible Markov process with state space S, there exists a steady state probability distribution

$$\{\pi_k, x_k \in S\}$$

This distribution is the same as the limiting or long term probability distribution:

$$\pi_k = \lim_{t \to \top} \Pr(X(t) = x_k \mid X(0) = x_0)$$

Existence of a steady state probability distribution

For every time-homogeneous, finite, irreducible Markov process with state space S, there exists a steady state probability distribution

$$\{\pi_k, x_k \in S\}$$

This distribution is the same as the limiting or long term probability distribution:

$$\pi_k = \lim_{t \to \top} \Pr(X(t) = x_k \mid X(0) = x_0)$$

This distribution is reached when the initial state no longer has any influence.

Probability flux

In steady state, $\boldsymbol{\pi_i}$ is the proportion of time that the process spends in state x_i.

Probability flux

In steady state, $\boldsymbol{\pi_i}$ is the proportion of time that the process spends in state x_i.

Recall q_{ij} is the instantaneous probability that the model makes a transition from state x_i to state x_j.

Probability flux

In steady state, $\boldsymbol{\pi}_\mathbf{i}$ is the proportion of time that the process spends in state x_i.

Recall q_{ij} is the instantaneous probability that the model makes a transition from state x_i to state x_j.

Thus, in an instant of time, the probability that a transition will occur from state x_i to state x_j is the probability that the model was in state x_i, $\boldsymbol{\pi}_\mathbf{i}$, multiplied by the transition rate q_{ij}.

Probability flux

In steady state, $\boldsymbol{\pi_i}$ is the proportion of time that the process spends in state x_i.

Recall q_{ij} is the instantaneous probability that the model makes a transition from state x_i to state x_j.

Thus, in an instant of time, the probability that a transition will occur from state x_i to state x_j is the probability that the model was in state x_i, $\boldsymbol{\pi_i}$, multiplied by the transition rate q_{ij}.

This is called the probability flux from state x_i to state x_j.

Global balance equations

In steady state, equilibrium is maintained so for any state the total probability flux out is equal to the total probability flux into the state.

$$\boxed{\underbrace{\boldsymbol{\pi}_{\mathbf{i}} \times \sum_{x_j \in S, j \neq i} q_{ij}}_{\text{flux \textbf{out} of } x_i} = \underbrace{\sum_{x_j \in S, j \neq i} (\boldsymbol{\pi}_{\mathbf{j}} \times q_{ji})}_{\text{flux \textbf{into} } x_i}}$$

Global balance equations

In steady state, equilibrium is maintained so for any state the total probability flux out is equal to the total probability flux into the state.

$$\boxed{\underbrace{\boldsymbol{\pi_i} \times \sum_{x_j \in S, j \neq i} q_{ij}}_{\text{flux } \textbf{out} \text{ of } x_i} = \underbrace{\sum_{x_j \in S, j \neq i} (\boldsymbol{\pi_j} \times q_{ji})}_{\text{flux } \textbf{into } x_i}}$$

(If this were not true the distribution over states would change.)

Global balance equations

Recall that the diagonal elements of the infinitesimal generator matrix $\mathbf{Q}$ are the negative sum of the other elements in the row, i.e. $q_{ii} = -\sum_{x_j \in S, j \neq i} q_{ij}$.

Global balance equations

Recall that the diagonal elements of the infinitesimal generator matrix $\mathbf{Q}$ are the negative sum of the other elements in the row, i.e. $q_{ii} = -\sum_{x_j \in S, j \neq i} q_{ij}$.

We can use this to rearrange the flux balance equation to be:

$$\sum_{x_j \in S} \boldsymbol{\pi}_\mathbf{j} q_{ji} = 0.$$

Global balance equations

Recall that the diagonal elements of the infinitesimal generator matrix $\mathbf{Q}$ are the negative sum of the other elements in the row, i.e. $q_{ii} = -\sum_{x_j \in S, j \neq i} q_{ij}$.

We can use this to rearrange the flux balance equation to be:

$$\sum_{x_j \in S} \boldsymbol{\pi}_\mathbf{j} q_{ji} = 0.$$

Expressing the unknown values $\boldsymbol{\pi}_\mathbf{i}$ as a row vector $\boldsymbol{\pi}$, we can write this as a matrix equation:

$$\boxed{\boldsymbol{\pi}\ \mathbf{Q} = 0}$$

Normalising constant

The π_i are unknown — they are the values we wish to find.

Normalising constant

The $\boldsymbol{\pi_i}$ are unknown — they are the values we wish to find.

If there are N states in the state space, the global balance equations give us N equations in N unknowns.

Normalising constant

The $\boldsymbol{\pi_i}$ are unknown — they are the values we wish to find.

If there are N states in the state space, the global balance equations give us N equations in N unknowns.

However this collection of equations is irreducible.

Normalising constant

The $\boldsymbol{\pi}_\mathbf{i}$ are unknown — they are the values we wish to find.

If there are N states in the state space, the global balance equations give us N equations in N unknowns.

However this collection of equations is irreducible.

Fortunately, since $\{\boldsymbol{\pi}_\mathbf{i}\}$ is a probability distribution we also know that the normalisation condition holds:

$$\boxed{\sum_{x_i \in S} \boldsymbol{\pi}_\mathbf{i} = 1}$$

Normalising constant

The $\boldsymbol{\pi_i}$ are unknown — they are the values we wish to find.

If there are N states in the state space, the global balance equations give us N equations in N unknowns.

However this collection of equations is irreducible.

Fortunately, since $\{\boldsymbol{\pi_i}\}$ is a probability distribution we also know that the normalisation condition holds:

$$\boxed{\sum_{x_i \in S} \boldsymbol{\pi_i} = 1}$$

With these $n+1$ equations we can use standard linear algebra techniques to solve the equations and find the n unknowns, $\{\boldsymbol{\pi_i}\}$.

Example

- Consider a system with multiple CPUs, each with its own private memory, and one common memory which can be accessed only by one processor at a time.

Example

- Consider a system with multiple CPUs, each with its own private memory, and one common memory which can be accessed only by one processor at a time.
- The CPUs execute in private memory for a random time before issuing a common memory access request. Assume that this random time is exponentially distributed with parameter λ (the average time a CPU spends executing in private memory between two common memory access requests is $1/\lambda$).

Example

- Consider a system with multiple CPUs, each with its own private memory, and one common memory which can be accessed only by one processor at a time.
- The CPUs execute in private memory for a random time before issuing a common memory access request. Assume that this random time is exponentially distributed with parameter λ (the average time a CPU spends executing in private memory between two common memory access requests is $1/\lambda$).
- The common memory access duration is also assumed to be exponentially distributed, with parameter μ (the average duration of a common memory access is $1/\mu$).

Example

If the system has only one processor, it has only two states:

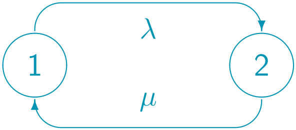
1. The processor is executing in its private memory;
2. The processor is accessing common memory.

Example

If the system has only one processor, it has only two states:

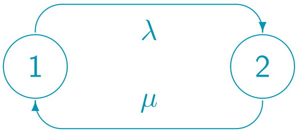
1. The processor is executing in its private memory;
2. The processor is accessing common memory.

The system behaviour can be modelled by a 2-state Markov process whose state transition diagram and generator matrix are as shown below:



$$\mathbf{Q} = \begin{pmatrix} -\lambda & \lambda \\ \mu & -\mu \end{pmatrix}$$

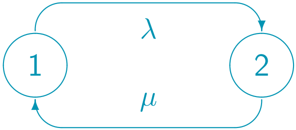
Example



$$\mathbf{Q} = \begin{pmatrix} -\lambda & \lambda \\ \mu & -\mu \end{pmatrix}$$

If we consider the probability flux in and out of state 1 we obtain: $\boldsymbol{\pi_1}\ \lambda = \boldsymbol{\pi_2}\mu$. Similarly, for state 2: $\boldsymbol{\pi_2}\ \mu = \boldsymbol{\pi_1}\lambda$.

Example

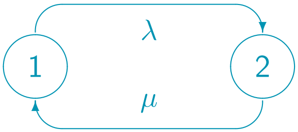


$$\mathbf{Q} = \begin{pmatrix} -\lambda & \lambda \\ \mu & -\mu \end{pmatrix}$$

If we consider the probability flux in and out of state 1 we obtain: $\boldsymbol{\pi_1}\ \lambda = \boldsymbol{\pi_2}\mu$. Similarly, for state 2: $\boldsymbol{\pi_2}\ \mu = \boldsymbol{\pi_1}\lambda$.

We know from the normalisation condition that: $\boldsymbol{\pi_1} + \boldsymbol{\pi_2} = 1$.

Example



$$\mathbf{Q} = \begin{pmatrix} -\lambda & \lambda \\ \mu & -\mu \end{pmatrix}$$

If we consider the probability flux in and out of state 1 we obtain: $\boldsymbol{\pi_1}\ \lambda = \boldsymbol{\pi_2}\mu$. Similarly, for state 2: $\boldsymbol{\pi_2}\ \mu = \boldsymbol{\pi_1}\lambda$.

We know from the normalisation condition that: $\boldsymbol{\pi_1} + \boldsymbol{\pi_2} = 1$.

Thus the steady state probability distribution is
$\boldsymbol{\pi} = \left(\frac{\mu}{\mu+\lambda}, \frac{\lambda}{\mu+\lambda}\right)$.

Example

$$\begin{tikzcd} 1 \arrow[rr, bend left, "\lambda"] & & 2 \arrow[ll, bend left, "\mu"] \end{tikzcd} \qquad \mathbf{Q} = \begin{pmatrix} -\lambda & \lambda \\ \mu & -\mu \end{pmatrix}$$

If we consider the probability flux in and out of state 1 we obtain: $\boldsymbol{\pi_1}\ \lambda = \boldsymbol{\pi_2}\mu$. Similarly, for state 2: $\boldsymbol{\pi_2}\ \mu = \boldsymbol{\pi_1}\lambda$.

We know from the normalisation condition that: $\boldsymbol{\pi_1} + \boldsymbol{\pi_2} = 1$.

Thus the steady state probability distribution is
$\boldsymbol{\pi} = \left(\frac{\mu}{\mu+\lambda}, \frac{\lambda}{\mu+\lambda}\right)$.

From this we can deduce, for example, that the probability that the processor is executing in private memory is $\mu/(\mu+\lambda)$.

Solving the global balance equations

In general our systems of equations will be too large to contemplate solving them by hand, so we want to be able to take advantage of linear algebra packages which can solve matrix equations of the form $\mathbf{Ax} = \mathbf{b}$, where $\mathbf{A}$ is an $N \times N$ matrix, $\mathbf{x}$ is a column vector of N unknowns, and $\mathbf{b}$ is a column vector of N values.

Solving the global balance equations

First we must resolve two problems:

1. Our global balance equation is expressed in terms of a row vector of unknowns $\boldsymbol{\pi}$, $\boldsymbol{\pi}\ \mathbf{Q} = 0$: the unknowns.

Solving the global balance equations

First we must resolve two problems:

1 Our global balance equation is expressed in terms of a row vector of unknowns $\boldsymbol{\pi}$, $\boldsymbol{\pi}\ \mathbf{Q} = 0$: the unknowns.

This problem is resolved by transposing the equation, i.e. $\mathbf{Q}^T\boldsymbol{\pi} = 0$, where the right hand side is now a column vector of zeros, rather than a row vector.

Solving the global balance equations

2. We must eliminate the redundancy in the global balance equations and add in the normalisation condition.

Solving the global balance equations

2 We must eliminate the redundancy in the global balance equations and add in the normalisation condition.

We replace one of the global balance equations by the normalisation condition. In $\mathbf{Q}^T$ this corresponds to replacing one row by a row of 1's. We usually choose the last row and denote the modified matrix $\mathbf{Q}_N^T$.

Solving the global balance equations

2 We must eliminate the redundancy in the global balance equations and add in the normalisation condition.

We replace one of the global balance equations by the normalisation condition. In $\mathbf{Q}^T$ this corresponds to replacing one row by a row of 1's. We usually choose the last row and denote the modified matrix $\mathbf{Q}_N^T$.

We must also make the corresponding change to the "solution" vector $\mathbf{0}$, to be a column vector with 1 in the last row, and zeros everywhere else. We denote this vector, $\mathbf{e}_N$.

Solving the global balance equations

2 We must eliminate the redundancy in the global balance equations and add in the normalisation condition.

We replace one of the global balance equations by the normalisation condition. In $\mathbf{Q}^T$ this corresponds to replacing one row by a row of 1's. We usually choose the last row and denote the modified matrix $\mathbf{Q}_N^T$.

We must also make the corresponding change to the "solution" vector $\mathbf{0}$, to be a column vector with 1 in the last row, and zeros everywhere else. We denote this vector, $\mathbf{e}_N$.

Now we can use any linear algebra solution package, such as `maple` or `xmaple` to solve the resulting equation:

$$\mathbf{Q}_N^T \boldsymbol{\pi} = \mathbf{e}_N$$

Example

Consider the two-processor version of the multiprocessor with processors A and B.

Example

Consider the two-processor version of the multiprocessor with processors A and B.

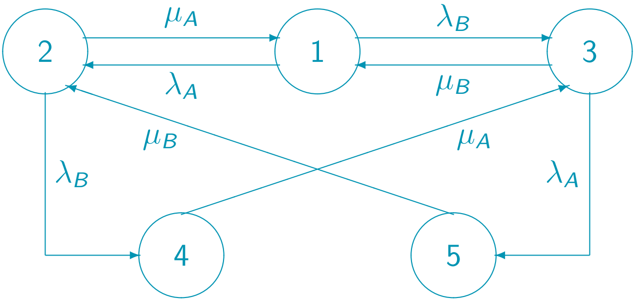
We assume that the processors have different timing characteristics, the private memory access of A being governed by an exponential distribution with parameter λ_A, the common memory access of B being governed by an exponential distribution with parameter μ_B, etc.

Example: state space

Now the state space becomes:

1. A and B both executing in their private memories;
2. B executing in private memory, and A accessing common memory;
3. A executing in private memory, and B accessing common memory;
4. A accessing common memory, B waiting for common memory;
5. B accessing common memory, A waiting for common memory;

Example: state space



Example: generator matrix

$$\mathbf{Q} = \begin{pmatrix} -(\lambda_A + \lambda_B) & \lambda_A & \lambda_B & 0 & 0 \\ \mu_A & -(\mu_A + \lambda_B) & 0 & \lambda_B & 0 \\ \mu_B & 0 & -(\mu_B + \lambda_A) & 0 & \lambda_A \\ 0 & 0 & \mu_A & -\mu_A & 0 \\ 0 & \mu_B & 0 & 0 & -\mu_B \end{pmatrix}$$

Example: modified generator matrix

$$\mathbf{Q}_N^T = \begin{pmatrix} -(\lambda_A + \lambda_B) & \mu_A & \mu_B & 0 & 0 \\ \lambda_A & -(\mu_A + \lambda_B) & 0 & 0 & \mu_B \\ \lambda_B & 0 & -(\mu_B + \lambda_A) & \mu_A & 0 \\ 0 & \lambda_B & 0 & -\mu_A & 0 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix}$$

Example: steady state probability distribution

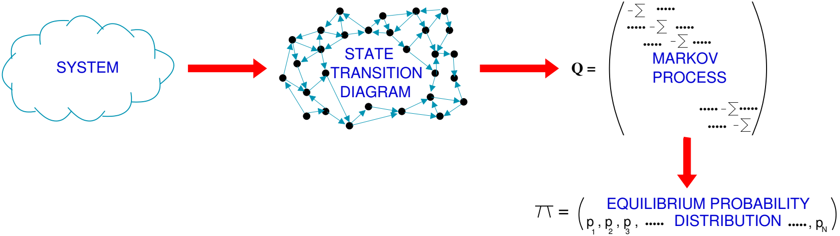
If we choose the following values for the parameters:

$$\lambda_A = 0.05 \qquad \lambda_B := 0.1 \qquad \mu_A = 0.02 \qquad \mu_B = 0.05$$

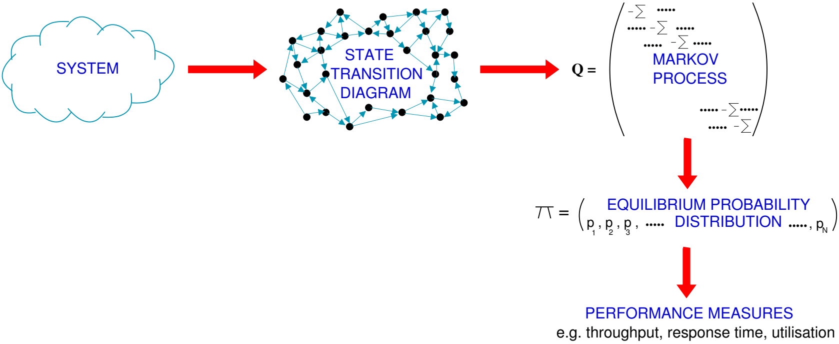
solving the matrix equation, and rounding figures to 4 significant figures, we obtain:

$$\boldsymbol{\pi} = (0.0693,\ 0.0990,\ 0.1683,\ 0.4951,\ 0.1683)$$

Deriving Performance Measures



Deriving Performance Measures



Deriving Performance Measures

Broadly speaking, there are three ways in which performance measures can be derived from the steady state distribution of a Markov process.

Deriving Performance Measures

Broadly speaking, there are three ways in which performance measures can be derived from the steady state distribution of a Markov process.

These different methods can be thought of as corresponding to different types of measure:

- state-based measures, e.g. utilisation;

Deriving Performance Measures

Broadly speaking, there are three ways in which performance measures can be derived from the steady state distribution of a Markov process.

These different methods can be thought of as corresponding to different types of measure:

- state-based measures, e.g. utilisation;
- rate-based measures, e.g. throughput;

Deriving Performance Measures

Broadly speaking, there are three ways in which performance measures can be derived from the steady state distribution of a Markov process.

These different methods can be thought of as corresponding to different types of measure:

- state-based measures, e.g. utilisation;
- rate-based measures, e.g. throughput;
- other measures which fall outside the above categories, e.g. response time.

State-based measures

State-based measures correspond to the probability that the model is in a state, or a subset of states, which satisfy some condition.

State-based measures

State-based measures correspond to the probability that the model is in a state, or a subset of states, which satisfy some condition.

For example, utilisation will correspond to those states where a resource is in use.

State-based measures

State-based measures correspond to the probability that the model is in a state, or a subset of states, which satisfy some condition.

For example, utilisation will correspond to those states where a resource is in use.

If we consider the multiprocessor example, the utilisation of the common memory, U_{mem}, is the total probability that the model is in one of the states in which the common memory is in use:

$$U_{mem} = \boldsymbol{\pi_2} + \boldsymbol{\pi_3} + \boldsymbol{\pi_4} + \boldsymbol{\pi_5} = 93.07\%$$

State-based measures

Other examples of state-based measures are idle time, or the number of jobs in a system.

State-based measures

Other examples of state-based measures are idle time, or the number of jobs in a system.

Some measures such as the number of jobs will involve a weighted sum of steady state probabilities, weighted by the appropriate value (expectation).

State-based measures

Other examples of state-based measures are idle time, or the number of jobs in a system.

Some measures such as the number of jobs will involve a weighted sum of steady state probabilities, weighted by the appropriate value (expectation).

For example, if we consider jobs waiting for the common memory to be queued in that subsystem, then the average number of jobs in the common memory, N_{mem}, is:

$$N_{mem} = (1 \times \boldsymbol{\pi_2}) + (1 \times \boldsymbol{\pi_3}) + (2 \times \boldsymbol{\pi_4}) + (2 \times \boldsymbol{\pi_5}) = 1.594$$

Rate-based measures

Rate-based measures are those which correspond to the predicted rate at which some event occurs.

Rate-based measures

Rate-based measures are those which correspond to the predicted rate at which some event occurs.

This will be the product of the rate of the event, and the probability that the event is enabled, i.e. the probability of being in one of the states from which the event can occur.

Example: rate-based measures

In order to calculate the throughput of the common memory, we need the average number of accesses from either processor which it satisfies in unit time.

Example: rate-based measures

In order to calculate the throughput of the common memory, we need the average number of accesses from either processor which it satisfies in unit time.

X_{mem} is thus calculated as:

$$X_{mem} = (\mu_A \times (\boldsymbol{\pi_2} + \boldsymbol{\pi_4})) + (\mu_B \times (\boldsymbol{\pi_3} + \boldsymbol{\pi_5})) = 0.0287$$

or, approximately one access every 35 milliseconds.

Other measures

The other measures are those which are neither rate-based or state-based.

Other measures

The other measures are those which are neither rate-based or state-based.

In these cases, we usually use one of the operational laws to derive the information we need, based on values that we have obtained from solution of the model.

Other measures

The other measures are those which are neither rate-based or state-based.

In these cases, we usually use one of the operational laws to derive the information we need, based on values that we have obtained from solution of the model.

For example, applying Little's Law to the common memory we see that

$$W_{mem} = N_{mem}/X_{mem} = 1.594/0.0287 = 55.54 \text{ milliseconds}$$

Assumptions

Stochastic Hypothesis

"The behaviour of a real system during a given period of time is characterised by the probability distributions of a stochastic process."

Assumptions

- All delays and inter-event times are exponentially distributed.

Assumptions

- All delays and inter-event times are exponentially distributed.
- (This will often not fit with observations of real systems.)

Assumptions

- All delays and inter-event times are exponentially distributed.
- (This will often not fit with observations of real systems.)
- We make the assumption because of the nice mathematical properties of the exponential distribution, and because it is the only distribution giving us a Markov process.

Assumptions

- All delays and inter-event times are exponentially distributed.
- (This will often not fit with observations of real systems.)
- We make the assumption because of the nice mathematical properties of the exponential distribution, and because it is the only distribution giving us a Markov process.
- Plus only a single parameter to be fitted (the rate), which can be easily derived from observations of the average duration.

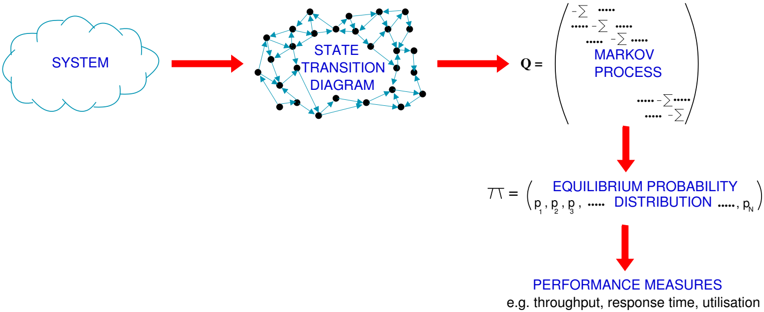
Assumptions

- All delays and inter-event times are exponentially distributed.
- (This will often not fit with observations of real systems.)
- We make the assumption because of the nice mathematical properties of the exponential distribution, and because it is the only distribution giving us a Markov process.
- Plus only a single parameter to be fitted (the rate), which can be easily derived from observations of the average duration.
- The Markov/memoryless assumption that future behaviour is only dependent on the current state, not on the past history is a reasonable assumption for computer and communication systems, if we choose our states carefully.

Assumptions

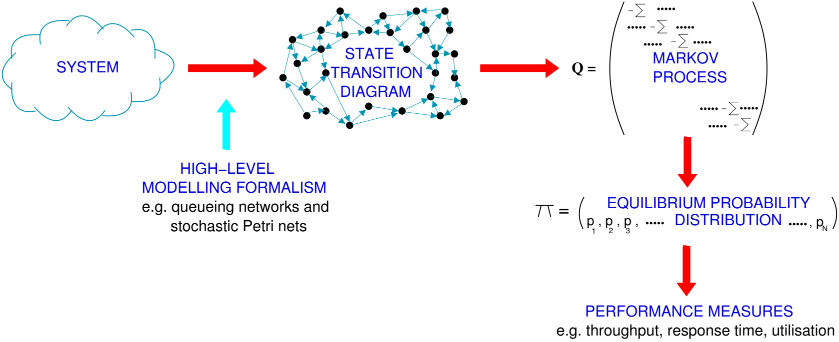
- All delays and inter-event times are exponentially distributed.
- (This will often not fit with observations of real systems.)
- We make the assumption because of the nice mathematical properties of the exponential distribution, and because it is the only distribution giving us a Markov process.
- Plus only a single parameter to be fitted (the rate), which can be easily derived from observations of the average duration.
- The Markov/memoryless assumption that future behaviour is only dependent on the current state, not on the past history is a reasonable assumption for computer and communication systems, if we choose our states carefully.
- We generally assume that the Markov process is finite, time homogeneous and irreducible.

Difficulties of working with Markov processes



Whilst Markov process-based modelling has many advantages, working directly in terms of the state transition diagram or infinitesimal generator matrix is at best time-consuming and error prone, and often simply infeasible.

Difficulties of working with Markov processes



For this reason various high level modelling formalisms have been introduced to make the job of constructing the state transition diagram and/or infinitesimal generator matrix easier.

Outline

The PEPA project

- The PEPA project started in Edinburgh in 1991.

The PEPA project

- The PEPA project started in Edinburgh in 1991.
- It was motivated by problems encountered when carrying out performance analysis of large computer and communication systems, based on numerical analysis of Markov processes.

The PEPA project

- The PEPA project started in Edinburgh in 1991.
- It was motivated by problems encountered when carrying out performance analysis of large computer and communication systems, based on numerical analysis of Markov processes.
- Process algebras offered a compositional description technique supported by apparatus for formal reasoning.

The PEPA project

- The PEPA project started in Edinburgh in 1991.
- It was motivated by problems encountered when carrying out performance analysis of large computer and communication systems, based on numerical analysis of Markov processes.
- Process algebras offered a compositional description technique supported by apparatus for formal reasoning.
- Performance Evaluation Process Algebra (PEPA) sought to address these problems by the introduction of a suitable process algebra.

The PEPA project

- The PEPA project started in Edinburgh in 1991.
- It was motivated by problems encountered when carrying out performance analysis of large computer and communication systems, based on numerical analysis of Markov processes.
- Process algebras offered a compositional description technique supported by apparatus for formal reasoning.
- Performance Evaluation Process Algebra (PEPA) sought to address these problems by the introduction of a suitable process algebra.
- We have sought to investigate and exploit the interplay between the process algebra and the continuous time Markov chain (CTMC).

Performance Evaluation Process Algebra

- PEPA (Performance Evaluation Process Algebra) is a high-level modelling language for distributed systems. It can be used to develop models of existing systems (abstraction) or designs for proposed ones (specification).
- PEPA can capture performance information in a process algebra setting. It is a stochastic process algebra.
- For technical details the definitive reference is *A Compositional Approach to Performance Modelling*, Hillston, Cambridge University Press, 1996.

Strengths of stochastic process algebras

SPAs have strengths in the areas of semantic definition, inherent compositionality and the existence of important equivalence relations (including bisimulation). This relation provides the basis for aggregation of PEPA models.

Terminology

The components in a PEPA model engage, cooperatively or individually, in activities.

Terminology

The components in a PEPA model engage, cooperatively or individually, in activities.

Each activity has an action type which corresponds to the actions of the system being modelled.

Terminology

The components in a PEPA model engage, cooperatively or individually, in activities.

Each activity has an action type which corresponds to the actions of the system being modelled.

To represent unimportant or unknown actions there is a distinguished action type, τ.

Quantitative aspects

Every activity in PEPA has an associated activity rate which may be any positive real number, or the distinguished symbol "$\top$", meaning unspecified, read as 'top'.

Quantitative aspects

Every activity in PEPA has an associated activity rate which may be any positive real number, or the distinguished symbol "$\top$", meaning unspecified, read as 'top'.

Components and activities are primitives. PEPA also provides a small set of combinators.

PEPA syntax

$$
\begin{array}{lcll}
S & ::= & (\alpha, r).S & \text{(prefix)} \\
  & | & S_1 + S_2 & \text{(choice)} \\
  & | & X & \text{(variable)} \\
C & ::= & C_1 \underset{L}{\bowtie} C_2 & \text{(cooperation)} \\
  & | & C \,/\, L & \text{(hiding)} \\
  & | & S & \text{(sequential)}
\end{array}
$$

PEPA: informal semantics (sequential sublanguage)

$(\alpha, r).S$

The activity (α, r) takes time Δt (drawn from the exponential distribution with parameter r).

$S_1 + S_2$

In this choice either S_1 or S_2 will complete an activity first. The other is discarded.

PEPA: informal semantics (combinators)

$C_1 \underset{L}{\bowtie} C_2$

All activities of C_1 and C_2 with types in L are shared: others remain individual.
NOTATION: write $C_1 \parallel C_2$ if L is empty.

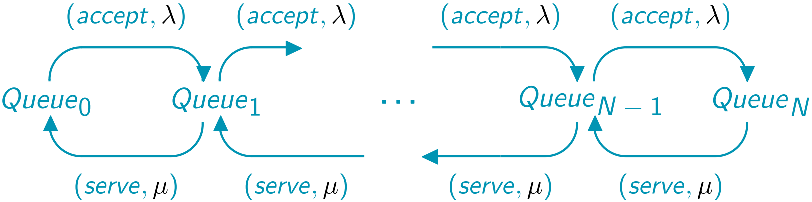
$C \,/\, L$

Activities of C with types in L are hidden (τ type activities) to be thought of as internal delays.

Example: M/M/1/N/N queue

$$
\begin{aligned}
Arrival_0 &\stackrel{def}{=} (accept, \lambda).Arrival_1 \\
Arrival_i &\stackrel{def}{=} (accept, \lambda).Arrival_{i+1} + (serve, \top).Arrival_{i-1} \\
& \qquad\qquad\qquad\qquad\qquad\qquad\qquad (0 < i < N) \\
Arrival_N &\stackrel{def}{=} (serve, \top).Arrival_{N-1} \\
Server &\stackrel{def}{=} (serve, \mu).Server
\end{aligned}
$$

Example: M/M/1/N/N queue



$$Queue_i \equiv Arrival_i \underset{\{serve\}}{\bowtie} Server$$